M3P

1.0.4

Generated by Doxygen 1.5.6

Tue Dec 22 12:25:48 2009

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 Metric Class Reference

Metric Class which captures the design metric concept.

`#include <metric.h>`

Public Member Functions

- Metric ()
  Metric default constructor.
- Metric (const string &rname, const string &rtype, const string &runit)
  Metric constructor.
- Metric (Metric const &metric)
  Metric copy constructor.
- ~Metric ()
  Metric default destructor.
- string getName () const
  Metric name acccesor.
- string getType () const
  Metric type acccesor.
- string getUnit () const
  Metric unit acccesor.
- unsigned long int getIntegerValue () const
  Metric integer value acccesor.
- long double getFloatValue () const
  Metric real value acccesor.

- void nameToLower ()

  Metric object name property character case conversion function to lower case.

- void setName (const string &rname)

  Metric name setting method.

- void setType (const string &rtype)

  Metric type setting method.

- void setUnit (const string &runit)

  Metric unit setting method.

- void setValue (const unsigned long int &rvalue)

  Metric integer value setting method.

- void setValue (const long double &rvalue)

  Metric floating point (real) value setting method.

- Metric & operator= (const Metric &m)

  Metric assignment operator.

Friends

- ostream & operator<< (ostream &os, const Metric &m)

  XML output stream operator (Metric object XML serializer).

- ostream & operator<< (ostream &os, const Metric *m)

  XML output stream operator (Metric object XML serializer).

2.1.1 Detailed Description

Metric Class which captures the design metric concept.

Author:

Gerardo de Miguel González

Version:

1.0

Date:

June 2008

2.1.2 Constructor & Destructor Documentation

2.1.2.1 Metric::Metric ()

Metric default constructor.

Returns:

A Metric Object

2.1.2.2 Metric::Metric (const string & *rname*, const string & *rtype*, const string & *runit*)

Metric constructor.

Parameters:

rname metric name (p.e execution_time)
rtype metric type (p.e float)
runit metric unit (p.e seconds)

Returns:

A Metric object

2.1.2.3 Metric::Metric (Metric const & *m*)

Metric copy constructor.

Parameters:

m a reference to a Metric object

Returns:

A Metric object

2.1.3 Member Function Documentation

2.1.3.1 string Metric::getName () const

Metric name acccesor.

Returns:

A string object which holds the name property content of the Metric object

2.1.3.2 string Metric::getType () const

Metric type acccesor.

Returns:

A string object which holds the type property content of the Metric object

2.1.3.3 string Metric::getUnit () const

Metric unit acccesor.

Returns:

A string object which holds the unit property content of the Metric object

2.1.3.4 unsigned long int Metric::getIntegerValue () const

Metric integer value acccesor.

Returns:

An unsigned integer number which holds the integer value of the Metric object

2.1.3.5 long double Metric::getFloatValue () const

Metric real value acccesor.

Returns:

A floating point number which holds the real value of the Metric object

2.1.3.6 void Metric::setName (const string & *rname*)

Metric name setting method.

Parameters:

rname a constant reference to a string object holding a name to set

2.1.3.7 void Metric::setType (const string & *rtype*)

Metric type setting method.

Parameters:

rtype a constant reference to a string object holding a type to set

2.1.3.8 void Metric::setUnit (const string & *runit*)

Metric unit setting method.

Parameters:

runit a constant reference to a string object holding a unit to set

2.1.3.9 void Metric::setValue (const unsigned long int & *rvalue*)

Metric integer value setting method.

Parameters:

rvalue a constant reference to an integer number holding a value to be set

2.1.3.10 void Metric::setValue (const long double & *rvalue*)

Metric floating point (real) value setting method.

Parameters:

rvalue a constant reference to floating-point number holding a value to be set

2.1.3.11 Metric & Metric::operator= (const Metric & *m*)

Metric assignment operator.

Parameters:

m a constant reference to a Metric object (rvalue)

Returns:

a reference to a Metric object (lvalue)

2.1.4 Friends And Related Function Documentation

2.1.4.1 ostream& operator<< (ostream & *os*, const Metric & *m*) [friend]

XML output stream operator (Metric object XML serializer).

Parameters:

os an output stream object reference

m a constant Metric object reference

Returns:

a reference to an output stream object

2.1.4.2 ostream& operator<< (ostream & *os*, const Metric * *m*) [friend]

XML output stream operator (Metric object XML serializer).

Parameters:

os an output stream object reference

m a constant Metric object pointer

Returns:

a reference to an output stream object

The documentation for this class was generated from the following files:

- metric.h
- metric.cpp

2.2 MetricReader Class Reference

Metric (expat + expatmm) Parser Class.

`#include <parse.h>`

Inherits expatmm::ExpatXMLParser.

Public Member Functions

- MetricReader ()

  MetricReader default constructor.

- MetricReader (const char *xmlFile)

  MetricReader constructor (from a XML file source).

- ~MetricReader ()

  Metric default destructor.

- bool writeXMLFile (const char *xmlOutputFile)

  writes an XML file with the Metric objects as specified in MULTICUBE

- bool writeXMLErrorFile (const char *xmlOutputFile, const char *reason, const char *kind)

  writes an XML error file as specified in MULTICUBE

- Metric getMetric ()

  Metric object acccesor Destructive retrieval of a Metric object from the Stack.

- void setMetric (Metric &m)

  Metric object setting method Insertion of a Metric object on the Stack.

- bool isMetricLeft () const

  checks whether there is some Metric object left in the Stack

- int size () const

  size of the Stack holding the Metric objects

Protected Member Functions

- virtual ssize_t read_block (void)

  expatmm (expat C++ wrapper) read_block method implementation Read the XML source (i.e a file) getting blocks of characters from a predefined block size which are then parsed and reported as events (SAX parsing style) which are handled with the start and end tag event handlers

- virtual void StartElement (const XML_Char *name, const XML_Char **attrs)

  StartElement Handler (SAX Parsing style) which triggers after parsing a starting tag in the XML file The data got from the attribute/value pairs is used to build a Metric object and the Metric object is pushed into a Stack (queued) container.

- virtual void EndElement (const XML_Char *name)

EndElement Handler (SAX Parsing style) which triggers after parsing an ending tag in the XML file.

2.2.1 Detailed Description

Metric (expat + expatmm) Parser Class.

Author:

Gerardo de Miguel González

Version:

1.0

Date:

June 2008

2.2.2 Constructor & Destructor Documentation

2.2.2.1 MetricReader::MetricReader ()

MetricReader default constructor.

Returns:

A MetricReader Object

Warning:

The default constructor is disabled declaring it private

2.2.2.2 MetricReader::MetricReader (const char * *xmlFile*)

MetricReader constructor (from a XML file source).

Parameters:

xmlFile XML file name where the MetricReader object is built from

Returns:

A MetricReader object

2.2.3 Member Function Documentation

2.2.3.1 bool MetricReader::writeXMLFile (const char * *xmlOutputFile*)

writes an XML file with the Metric objects as specified in MULTICUBE

Parameters:

xmlOutputFile constant char array holding the name of the XML output file

Returns:

A boolean result 'True' if the writing process has been successful

2.2.3.2 bool MetricReader::writeXMLErrorFile (const char * *xmlOutputFile*, const char * *reason*, const char * *kind*)

writes an XML error file as specified in MULTICUBE

Parameters:

xmlOutputFile constant char array holding the name of the XML output file

reason a word description of the exception or error reported

kind the severity grade of the error reported (i.e fatal)

Returns:

A boolean result 'True' if the writing process has been successful

2.2.3.3 Metric MetricReader::getMetric ()

Metric object acccesor Destructive retrieval of a Metric object from the Stack.

Returns:

A Metric object

2.2.3.4 void MetricReader::setMetric (Metric & *m*)

Metric object setting method Insertion of a Metric object on the Stack.

Parameters:

m the Metric object that is going to be pushed into the Stack

2.2.3.5 bool MetricReader::isMetricLeft () const

checks whether there is some Metric object left in the Stack

Returns:

A boolean result 'True' if there is some Metric object left

2.2.3.6 int MetricReader::size () const

size of the Stack holding the Metric objects

Returns:

the number of Metric objects which are in the Stack

2.2.3.7 ssize_t MetricReader::read_block (void) [protected, virtual]

expatmm (expat C++ wrapper) read_block method implementation Read the XML source (i.e a file) getting blocks of characters from a predefined block size which are then parsed and reported as events (SAX parsing style) which are handled with the start and end tag event handlers

Returns:

the number of blocks read or '-1' if there is some error

2.2.3.8 void MetricReader::StartElement (const XML_Char * *name*, const XML_Char ** *attrs*) [protected, virtual]

StartElement Handler (SAX Parsing style) which triggers after parsing a starting tag in the XML file The data got from the attribute/value pairs is used to build a Metric object and the Metric object is pushed into a Stack (queued) container.

Parameters:

name constant char array which holds the tag's name which is parsed

attrs constant Nx2 array which holds the attribute/value pairs within the XML tag which is parsed

Returns:

2.2.3.9 void MetricReader::EndElement (const XML_Char * *name*) [protected, virtual]

EndElement Handler (SAX Parsing style) which triggers after parsing an ending tag in the XML file.

Parameters:

name constant char array which holds the tag's name which is parsed

Returns:

The documentation for this class was generated from the following files:

- parse.h
- parse.cpp

Index